Introduction To Data Structures

This section introduces the concept of a data structure as well as providing the details of a specific example: a list.

What Is A Data Structure

- A composite type that has a set of basic operations (e.g., display elements of a list) that may be performed on instances of that type.
  - It can be accessed as a whole (e.g., pass the entire list as a parameter to a function).
  - Individual elements can also be accessed (e.g., update the value for a single list element).
- The type may be a built-in part of the programming language
  - e.g., lists are included with the Python language.
- The type may also be defined by the programmer inside a program (for languages which don't include this composite type)

```
class List
{
    :   :
}
```

What Is A Data Structure (2)

- In some cases the data structure may only be partially implemented as part of the language, some operations must be manually written by the programmer.
- Example: The ability to add an element to a list has been implemented as a function in Python.

```
aGrid = []              # Creates an empty list
aGrid.append (12)       # Adds a number to the end of the list
```

- In a language such as 'C' a list is implemented as an array but the operation to add elements to the end of the list must be written by the programmer.
- Moral: when choosing a programming language look for built-in support for key features.

Lists

- Lists are a type of data structure (one of the simplest and most commonly used).
  - e.g., grades for a lecture can be stored in the form of a list
- List operations: creation, adding new elements, searching for elements, removing existing elements, modifying elements, display elements, sorting elements, deleting the entire list).
- List implementation in Java: array, linked list.

Arrays

- An array of 'n' elements will have an index of zero for the first element up to index (n-1) for the last element.
- The array index is an integer and indicates which element to access (excluding the index and just providing the name of the list means that the program is operating on the entire list).
- Similar to objects, arrays employ dynamic memory allocation (the name of the array is actually a reference to the array).
- Many utility methods exist.
- Several error checking mechanisms are available.

Arrays

- An array of 'n' elements will have an index of zero for the first element up to index (n-1) for the last element.
- The array index is an integer and indicates which element to access (excluding the index and just providing the name of the list means that the program is operating on the entire list).
- **Similar to objects, arrays employ dynamic memory allocation (the name of the array is actually a reference to the array).**
- Many utility methods exist.
- Several error checking mechanisms are available.

James Tam

Declaring Arrays

- Arrays in Java involve a reference to the array so creating an array requires two steps:
  1) Declaring a reference to the array
  2) Allocating the memory for the array

Declaring A Reference To An Array

- **Format:**

```
// The number of pairs of square brackets specifies the number of
// dimensions.
<type> [] <array name>;
```

- **Example:**

```
int [] arr;
int [][] arr;
```

Allocating Memory For An Array

- **Format:**

  <*array name*> = new <*array type*> [<*no elements*>];

- **Example:**

```
arr = new int [SIZE];
arr = new int [ROW SIZE][COLUMN SIZE];
```

(Both steps can be combined together):

```
int [] arr = new int[SIZE];
```


Arrays: An Example

- This example can be found in UNIX under:
  /home/233/examples/lists/firstExample

```
public class Driver
{
   public static void main (String [] args)
   {
      int i;
      int len;
      int [] arr;
```

Arrays: An Example

```
        Scanner in = new Scanner (System.in);
        System.out.print("Enter the number of array elements: ");
        len = in.nextInt ();
        arr = new int [len];
        System.out.println("Array Arr has " + arr.length + " elements.");
        for (i = 0; i < arr.length; i++)
        {
           arr[i] = i;
           System.out.println("Element[" + i + "]=" + arr[i]);
        }
     }
  }
```


Arrays

- An array of 'n' elements will have an index of zero for the first element up to index (n-1) for the last element.
- The array index is an integer and indicates which element to access (excluding the index and just providing the name of the list means that the program is operating on the entire list).
- Similar to objects, arrays employ dynamic memory allocation (the name of the array is actually a reference to the array).
- Many utility methods exist.
- **Several error checking mechanisms are available.**
  - Null array references
  - Array bounds checking

Using A Null Reference

```
int [] arr = null;
arr[0] = 1;
```

NullPointerException

Exceeding The Array Bounds

```
int [] arr  = new int [4];
int i;
for (i = 0; i <= 4; i++)
  arr[i] = i;
```

ArrayIndexOutOfBoundsException

(when i = 4)

List Operations: Arrays (Creation)

•Simply declare an array variable

<*array name*> = new <*array type*> [<*no elements*>];

List Operations: Arrays (Display)

•Unless it can be guaranteed that the list will always be full (unlikely) then some mechanism for determining that the end of the list has been reached is needed.

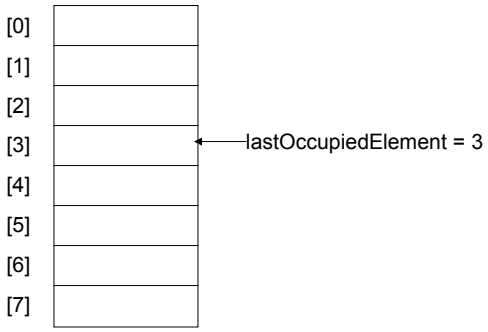
•If list elements cannot take on certain values then unoccupied list elements can be 'marked' with an invalid value.

•Example: grades

[0]	100
[1]	75
[2]	65
[3]	0
[4]	80
[5]	-1
[6]	-1
[7]	-1

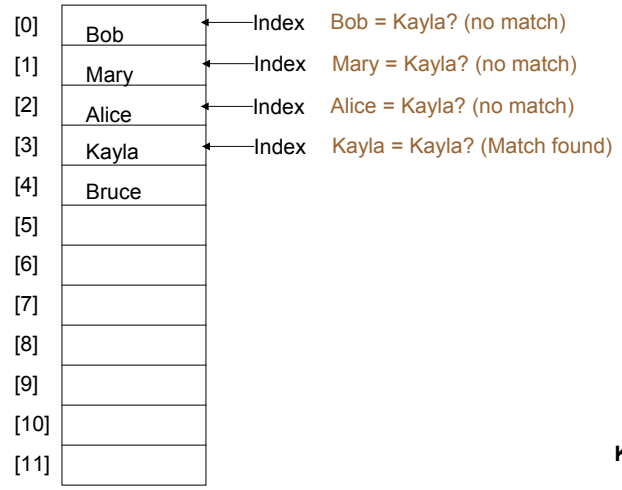
List Operations: Arrays (Display: 2)

- If list elements can't be marked then a special variable ("last" index) can be used to mark the last occupied element.
- Alternatively a special variable can also be used to mark the next element free.



List Operations: Arrays (Search)

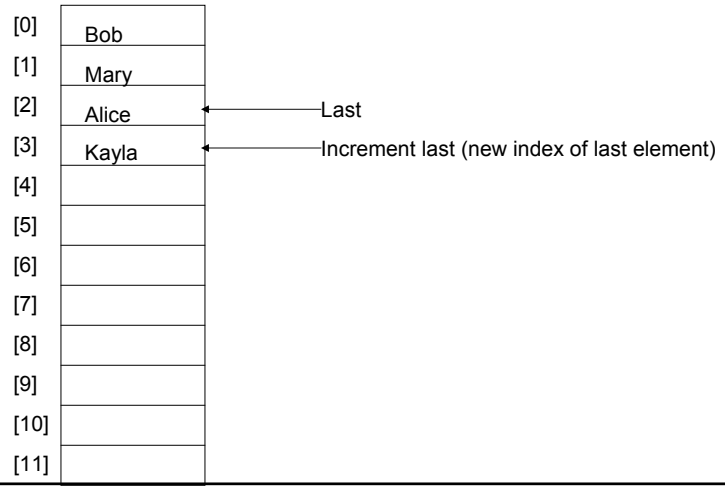
- Compare the item being searched for ("key") and compare vs. the list element.



List Operations: Arrays (Insertion)

•Insertion at the end.

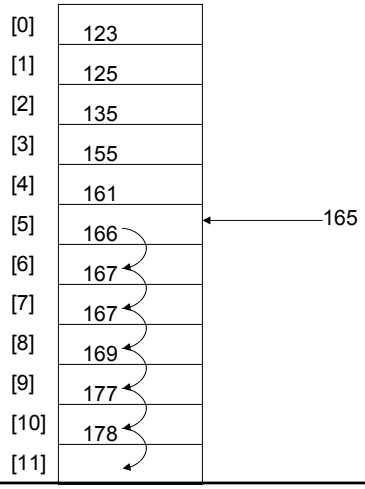
- Some mechanism is needed to either find or keep track of the last occupied element.



List Operations: Arrays (Insertion: 2)

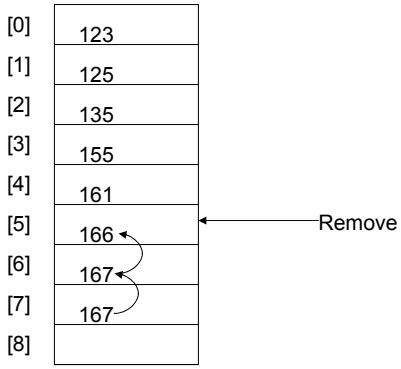
•In order insertion.

- Some mechanism is needed to find the insertion point (search).
- Elements may need to be shifted.



List Operations: Arrays (Removing Elements)

- A search is needed to find the element to remove.
- Depending upon the index of the element to be deleted, other elements may need to be shifted.



List Operations: Arrays (Destroying The Entire List)

- Recall that Java employs automatic garbage collection.
- Setting the reference to the array to null will eventually allow the array to be garbage collected.

  <*array name*> = null;

- Note: many languages do not employ automatic garbage collection and in those cases, either the entire array or each element must be manually de-allocated in memory.

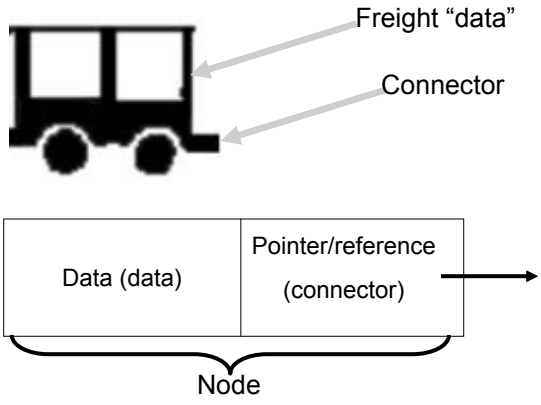
Memory Leak

- A technical term for programs that don't free up dynamically allocated memory.
- It can be serious problem because it may result in a drastic slowdown of a program.

Linked Lists

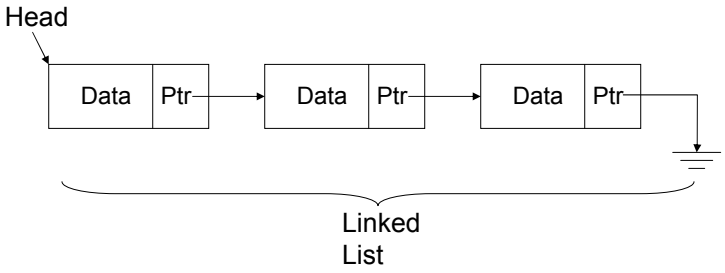
- An alternate implementation of a list.
- The program code is somewhat more complex but some operations are more efficient (e.g., additions and deletions don't require shifting of elements).
- Also linked lists tend to me more memory efficient that arrays.
  - The typical approach with an array implementation is to make the array larger than needed. (Unused elements are allocated in memory and the space is wasted).
  - With a linked list implementation, elements only take up space in memory as they're needed.

List Elements: Nodes



Linked Lists: Important Details

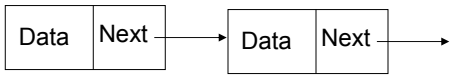
- Unlike arrays, many details must be manually and explicitly specified by the programmer: start of the list, connections between elements, end of the list.



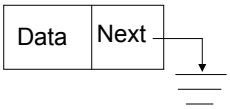
1 The approximate equivalent of a pointer (“ptr”) in Java is a reference.

More On Connections: The Next Pointer

- Because linked lists only create elements as needed a special marker is needed for the end of the list.
- The 'next' attribute of a node will either:
  - Contain a reference/address of the next node in the list.



  - Contain a null value.



List Operations: Linked Lists (Creation)

- After a type for the list has been declared then creating a new list requires that an instance be created and initialized.
- Example:

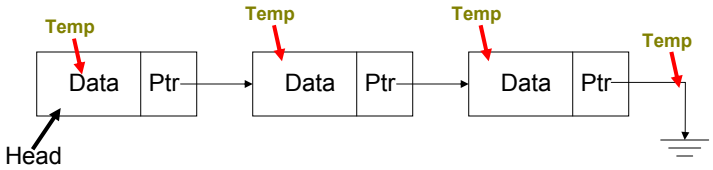
```
Node head = null;
```

List Operations: Linked Lists (Display)

- A temporary pointer/reference is used when successively displaying the elements of the list.
- When the temporary pointer is null, the end of the list has been reached.
- Pseudo code algorithm:

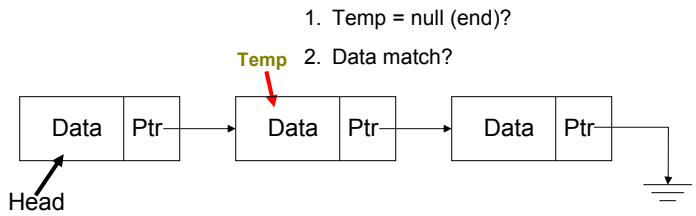
```
while (temp != null)
    display node
    temp = address of next node
```

- Graphical illustration of the algorithm:



List Operations: Linked Lists (Search)

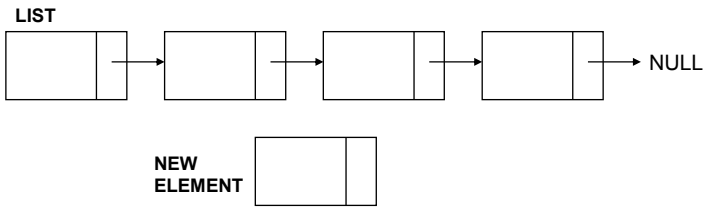
- The algorithm is similar to displaying list elements except that there must be an additional check to see if a match has occurred.



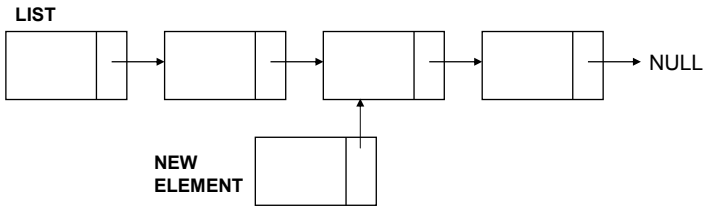
List Operations That Change List Membership

- These two operations (add/remove) change the number of elements in a list.
- The first step is to find the point in the list where the node is to be added or deleted (typically requires a search).
- Once the point in the list has been found, changing list membership is merely a reassignment of pointers/references.
  - Again: unlike the case with arrays, no shifting is needed.

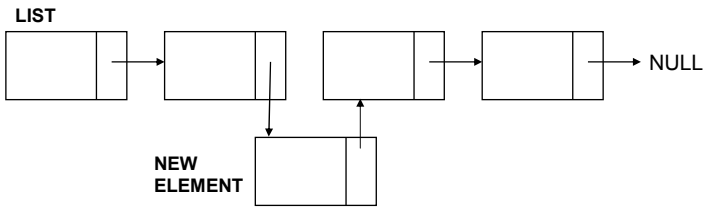
List Operations: Linked Lists (Insertion)



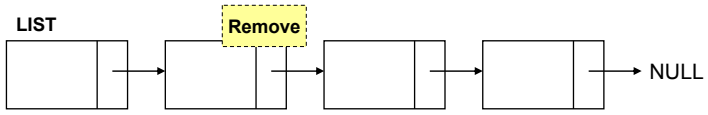
List Operations: Linked Lists (Insertion: 2)



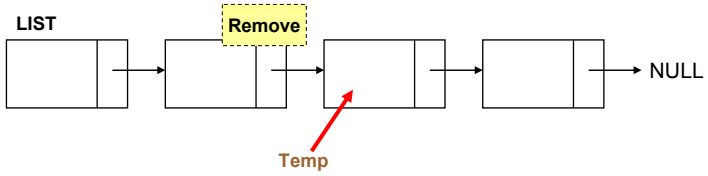
List Operations: Linked Lists (Insertion: 3)



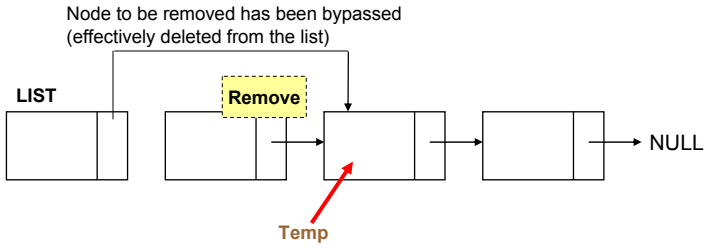
List Operations: Linked Lists (Removing Elements)



List Operations: Linked Lists (Removing Elements)



List Operations: Linked Lists (Removing Elements)



List Operations: Linked Lists (Destroying The Entire List)

- With linked lists removing an entire list is similar to how it's done with the array implementation.

```
head = null;
```

- Important reminder: many languages to not employ automatic garbage collection and in those cases each node must be manually de-allocated in memory (step through each element in the list and free up the memory but take care not to lose the connection with the rest of the list).

Arrays Of Objects (References)

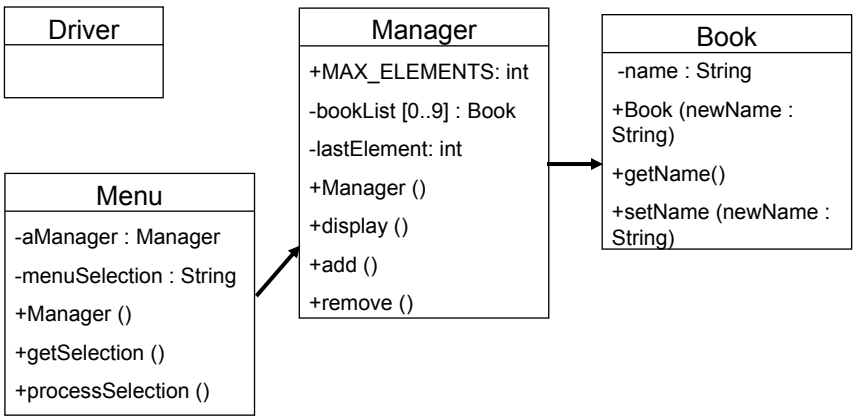
- An array of objects is actually an array of references to objects
  e.g., Foo [] arr = new Foo [4];

- The elements are initialized to null by default

arr[0].setNum(1); ← **NullPointerException**

Arrays Of References To Objects: An Example

- The complete example can be found in the directory
/home/233/examples/lists/secondExample



The Book Class

```
public class Book
{
   private String name;
   public Book (String aName)
   {
      setName(aName);
   }
   public void setName (String aName)
   {
      name = aName;
   }
   public String getName ()
   {
       return name;
   }
}
```


The Manager Class

```
public class Manager
{
   public final int MAX_ELEMENTS = 10;
   private Book [] bookList;
   private int lastElement;

   public Manager ()
   {
      bookList = new Book[MAX_ELEMENTS];
      int i;
      for (i = 0; i < MAX_ELEMENTS; i++)
      {
         bookList[i] = null;
      }
      lastElement = -1;
   }
```

The Manager Class (2)

```
public void display()
{
    int i;
    System.out.println("Displaying list");
    if (lastElement == -1)
        System.out.println("\tList is empty");
     for (i = 0; i <= lastElement; i++)
     {
         System.out.println("\tTitle No. " + (i+1) + ": "+ bookList[i].getName());
     }
}
```


The Manager Class (3)

```
public void add ()
{
    String newName;
    Scanner in;
    if ((lastElement+1) < MAX_ELEMENTS)
    {
        System.out.print("Enter a title for the book: ");
        in = new Scanner (System.in);
        newName = in.nextLine ();
        lastElement++;
        bookList[lastElement] = new Book(newName);
     }
     else
     {
         System.out.print("Cannot add new element: ");
         System.out.println("List already has " + MAX_ELEMENTS + "
         elements.");
     }
}
```

The Manager Class (4)

```
    public void remove ()
    {
        if (lastElement != -1)
        {
            bookList[lastElement] = null;
            lastElement--;
            System.out.println("Last element removed from list.");
         }
        else
        {
            System.out.println("List is already empty: Nothing to remove");
        }
    }
}
```


The Menu Class

```
public class Menu
{
   private Manager aManager;
   private String menuSelection;

   public Menu ()
   {
      aManager = new Manager ();
      menuSelection = null;
   }
```

The Menu Class (2)

```
public void display ()
{
    System.out.println("\n\nLIST MANAGEMENT PROGRAM: OPTIONS");
    System.out.println("\t(d)isplay list");
    System.out.println("\t(a)dd new element to end of list");
    System.out.println("\t(r)emove last element from the list");
    System.out.println("\t(q)uit program");
    System.out.print("Selection: ");
}

public void getSelection ()
{
    String newName;
    Scanner in = new Scanner (System.in);
    menuSelection = in.nextLine ();
}
```


The Menu Class (3)

```
public void processSelection ()
{
  do
  {
      display();
      getSelection();
      if (menuSelection.equals("d"))
          aManager.display ();
      else if (menuSelection.equals("a"))
          aManager.add ();
      else if (menuSelection.equals("r"))
          aManager.remove ();
      else if (menuSelection.equals("q"))
          System.out.println ("Quitting program.");
      else
         System.out.println("Please enter one of 'd','a','r' or 'q'");
   } while (!(menuSelection.equals("q")));
}
}
```

The Driver Class

```
public class Driver
{
   public static void main (String [] args)
   {
        Menu aMenu = new Menu ();
        aMenu.processSelection();
   }  // End of main.
}  // End of class Driver.
```


After This Section You Should Now Know

- What is a data structure
- How a data structure may be defined in Java
- Common list operations
- How the common list operations are implemented using arrays
- How a Java array employs dynamic memory allocation
- What is a memory leak
- How the common list operations are implemented using linked lists
- What are the advantages and disadvantages of implementing a list as an array vs. as a linked list
- How to implement a list with elements that are composite types (array of references)